Multirate time-stepping least squares shadowing method

By H. J. Bae AND P. Moin

1. Motivation and objectives

The recently developed least squares shadowing (LSS) method reformulates unsteady turbulent flow simulations to be well-conditioned time-domain boundary-value problems. We see from Wang *et al.* (2013) that the reformulation from LSS can enable scalable parallel in-time simulation of turbulent flows. It utilizes the large number of processors in high-performance machines in order to find a trajectory that satisfies the given governing equation effectively by relaxing the initial condition. This method can speed up the wall clock time of finding the solution by effectively parallelizing in the temporal domain as well as the spatial domain. However, the traditional LSS method was limited by the smallest time-step of the entire domain, and thus required solving of extremely large block tri-diagonal systems.

A LSS method with multirate time-stepping (MTLSS) was implemented to avoid the necessity to take small global time-steps (restricted by the largest value of the Courant-Friedrichs-Lewy (CFL) condition on the grid) and therefore resulting in substantially more efficient computations when the region that requires small time-steps are relatively small compared to the entire domain. In this approach, with MTLSS, the time-step can vary spatially and has to satisfy the CFL condition only locally, resulting in substantially more efficient computations when the region that requires small time-steps are relatively small compared to the entire domain.

The idea of multirate time-stepping is not new. This has been an effective way to compute for geometries with high discrepancy in CFL numbers (Constantinescu & Sandu 2007). Because the computation time step is no longer bound by the cell with the smallest CFL number, the computational gain is significant. Also, there have been studies by Seny *et al.* (2014) on how to utilize multirate time-stepping and how to achieve maximum computational gain by parallelizing. However, all these previous studies have been limited to parallel in space methods only. When coupled with a parallel in time methods like LSS, the computational gain is even more significant.

We will present the results of the multirate time-stepping LSS compared to a traditional LSS and discuss the problem memory requirements and computational savings.

2. Theoretical aspects

Consider a dynamical system governed by

$$\frac{du}{dt} = f(u; s), \tag{2.1}$$

where f is a nonlinear spatial operator, and s is a parameter. We can achieve a stable trajectory with a relaxed initial condition by splitting a perturbation into stable and unstable components and propagate their effects forward and backward in time, respectively.

Due to the shadowing lemma by Pilyugin (1999), we can achieve a stable trajectory. The least squares formulation is used to take advantage of the numerical stability. We seek to minimize the L_2-norm of the difference between the shadow trajectory u and some reference solution u_{ref}

$$\min_u \frac{1}{2}\int_0^T \|u(\tau(t)) - u_{ref}(t)\|^2 dt, \tag{2.2}$$

such that

$$\frac{du}{d\tau} = f(u; s + \delta s). \tag{2.3}$$

For forward sensitivity analysis, we are interested in the quantity $v \equiv \partial u/\partial s$, so by dividing by δs and taking $\delta s \to 0$,

$$\min_v \frac{1}{2}\int_0^T \|v(t)\|^2 dt, \tag{2.4}$$

such that

$$\frac{dv}{dt} = \frac{\partial f}{\partial u}v + \frac{\partial f}{\partial s} + \eta f, \tag{2.5}$$

where $\eta \equiv d\tau/dt - 1$.

We modify the formulation

$$v, \eta = \arg\min \frac{1}{2}\int_0^T \left(\|v\|^2 + \alpha^2\eta^2\right) dt, \tag{2.6}$$

such that

$$\frac{dv}{dt} = \frac{\partial f}{\partial u}v + \frac{\partial f}{\partial s} + \eta f. \tag{2.7}$$

The corresponding Karush-Kuhn-Tucker (KKT) system is

$$\frac{\partial w}{\partial t} = -\left(\frac{\partial f}{\partial u}\right)^* w - v, w(0) = w(T) = 0, \tag{2.8}$$

$$\alpha^2\eta = - < f, w >, \tag{2.9}$$

$$\frac{dv}{dt} = \frac{\partial f}{\partial u}v + \frac{\partial f}{\partial s} + \eta f, \tag{2.10}$$

which can be combined to form a single second-order equation

$$-\frac{d^2w}{dt^2} - \left[\frac{d}{dt}\left(\frac{\partial f}{\partial u}\right)^* - \frac{\partial f}{\partial u}\frac{d}{dt}\right] w + \left[\frac{\partial f}{\partial u}\left(\frac{\partial f}{\partial u}\right)^* + \frac{1}{\alpha}ff^*\right] w = \frac{\partial f}{\partial s}, \tag{2.11}$$

$$w(0) = w(T) = 0, \tag{2.12}$$

or, taking the Schur complement of the KKT system,

$$\left(\mathcal{B}\mathcal{B}^* + \mathcal{E}\mathcal{E}^*\right) w = b, \tag{2.13}$$

where the continuous linear operators $\mathcal{B}$ and $\mathcal{E}$ are defined as

$$\left(\mathcal{B}w\right)(t) = \left(\frac{d}{dt} - \left.\frac{\partial f}{\partial u}\right|_{u(t)}\right) w(t) \tag{2.14}$$

$$\left(\mathcal{E}w\right)(t) = \frac{1}{\alpha}f(u(t))w(t). \tag{2.15}$$

With a simple discretization, the system is very large for many problems of interest, and requires solving a linear system of $O(mn)$ equations, where m is the number of time steps and n is the number of dimensions or degrees of freedom of the system, both of which can be very large ($\sim 10^5$ or greater).

We perform the discretization in multiple time-phases in order to reduce the number of time steps in subdomains of the problem. In this paper, we deal with time steps that are power of 2 multiples of the minimal time step. That is, given the minimal time step Δt_{min}, all other time steps are of the form $\Delta t_m = 2^l \Delta t_{min}$, where l is an integer.

We first consider the case with two different time steps, Δt_f and $\Delta t_c = 2^l \Delta t_f$, and divide the spatial domain into two parts.

$$\Omega = \Omega_f \cup \Omega_c, \tag{2.16}$$

where, the domain Ω_f has time step Δt_f and Ω_c has time step Δt_c.

From now on u^f will refer to the vector corresponding to the fine region with time step increment Δt_f from $0 \to N$; $u^f_n = u(n\Delta t_f)|_{\Omega_f}$.

$$u^f \equiv [u^f_0, u^f_1, u^f_2, \cdots, u^f_{N-1}, u^f_N]^T, u^f \in R^{(N+1)|\Omega_f|}, \tag{2.17}$$

and u^c will refer to the vector corresponding to the coarse region with time step increment Δt_c from $0 \to 2^{-l}N$.

$$u^c \equiv [u^c_0, u^c_{2^l}, u^c_{2\cdot 2^l}, \cdots, u^c_{(N-1)\cdot 2^l}, u^c_{N\cdot 2^l}]^T, u^c \in R^{(2^{-l}N+1)|\Omega_c|}. \tag{2.18}$$

We then interpolate u^c such that $u^{c*} \in R^{(N+1)|\Omega_2|}$ and form

$$u \equiv [u_0, u_1, u_2, \cdots, u_{N-1}, u_N]^T, u \in R^{(N+1)|\Omega|}, \tag{2.19}$$

where u_i is formed by the vector u^f_i and u^{c*}_i. However, we can form the vector using $(N+1)|\Omega_f| + (2^{-l}N+1)|\Omega_c|$ elements instead of $(N+1)|\Omega|$ elements.

Thus, if we define

$$b_n \equiv \frac{1}{2}\left(\left.\frac{\partial f}{\partial s}\right|_{u_n} + \left.\frac{\partial f}{\partial s}\right|_{u_{n+1}} \right), \tag{2.20}$$

$$f_n \equiv \frac{1}{2}\left(f(u_n) + f(u_{n+1})\right), \tag{2.21}$$

$$A_n \equiv \left.\frac{\partial f}{\partial u}\right|_{\frac{u_n+u_{n+1}}{2}}, \tag{2.22}$$

then, the discretized operator becomes

$$(Bw)_n = \frac{w_{n+1} - w_n}{\Delta t_n} - A_n \frac{w_n + w_{n+1}}{2}, \tag{2.23}$$

$$(Ew)_n = \frac{1}{\alpha} f_n w_n, \tag{2.24}$$

such that the total system becomes

$$(BB^T + EE^T)w = b. \tag{2.25}$$

We will show below that BB^T and $EE^T \in R^{N(|\Omega_f|+2^{-l}|\Omega_c|)\times N(|\Omega_f|+2^{-l}|\Omega_c|)}$ for the multirate time-stepping method and these are symmetric block tridiagonal matrices with block size of $(2^l|\Omega_f| + |\Omega_c|) \times (2^l|\Omega_f| + |\Omega_c|)$, instead of $R^{N|\Omega|\times N|\Omega|}$ with block size

$|\Omega| \times |\Omega|$ as it previously was with a non-multirate time-stepping method. Also, $w, b \in R^{(N|\Omega_f|+2^{-l}N|\Omega_c|)}$.

2.1. *Reduction in problem size*

We can define the matrix B in a non-multirate LSS operation as a bi-diagonal matrix as seen below.

$$B = \begin{pmatrix} -\frac{I}{\Delta t_1} - \frac{A_1}{2} & \frac{I}{\Delta t_1} - \frac{A_1}{2} & & & \\ & -\frac{I}{\Delta t_2} - \frac{A_2}{2} & \frac{I}{\Delta t_2} - \frac{A_2}{2} & & \\ & & \ddots & \ddots & \\ & & & -\frac{I}{\Delta t_{N-1}} - \frac{A_{N-1}}{2} & \frac{I}{\Delta t_{N-1}} - \frac{A_{N-1}}{2} \end{pmatrix}. \tag{2.26}$$

Each I and $A_n \in R^{|\Omega|\times|\Omega|}$. We can permute these matrices so that the course and fine subdomains are bundled together. That is, we can order rows and columns of I and A_n so that we can organize it as follows.

$$\begin{aligned}(Bw)_n = &\begin{pmatrix} -\frac{I^{ff}}{\Delta t_n} - \frac{A_n^{ff}}{2} & -\frac{I^{cf}}{\Delta t_n} - \frac{A_n^{cf}}{2} \\ -\frac{I^{fc}}{\Delta t_n} - \frac{A_n^{fc}}{2} & -\frac{I^{cc}}{\Delta t_n} - \frac{A_n^{cc}}{2} \end{pmatrix} \begin{pmatrix} w_n^f \\ w_n^c \end{pmatrix} \\ &+ \begin{pmatrix} \frac{I^{ff}}{\Delta t_{n+1}} - \frac{A_{n+1}^{ff}}{2} & \frac{I^{cf}}{\Delta t_{n+1}} - \frac{A_{n+1}^{cf}}{2} \\ \frac{I^{fc}}{\Delta t_n} - \frac{A_n^{fc}}{2} & \frac{I^{cc}}{\Delta t_n} - \frac{A_n^{cc}}{2} \end{pmatrix} \begin{pmatrix} w_{n+1}^f \\ w_{n+1}^c \end{pmatrix}, \end{aligned} \tag{2.27}$$

where A^{ab} maps values in Ω_a to Ω_b.

Suppose $m \cdot 2^l \le n < (m+1) \cdot 2^l$, then w_n^c and w_{n+1}^c can be written as a linear sum of $w_{m\cdot 2^l}^c$ and $w_{(m+1)\cdot 2^l}^c$. Then, we can rewrite (2.27) as

$$\begin{aligned}(Bw)_n = &\begin{pmatrix} -\frac{I^{ff}}{\Delta t_n} - \frac{A_n^{ff}}{2} & -\frac{I^{cf}}{\Delta t_n} - \frac{A_n^{cf}}{2} \\ -\frac{I^{fc}}{\Delta t_n} - \frac{A_n^{fc}}{2} & -\frac{I^{cc}}{\Delta t_n} - \frac{A_n^{cc}}{2} \end{pmatrix} \begin{pmatrix} w_n^f \\ p_n \cdot w_{m\cdot 2^l}^c + (1-p_n) \cdot w_{(m+1)\cdot 2^l}^c \end{pmatrix} \\ + &\begin{pmatrix} \frac{I^{ff}}{\Delta t_{n+1}} - \frac{A_{n+1}^{ff}}{2} & \frac{I^{cf}}{\Delta t_{n+1}} - \frac{A_{n+1}^{cf}}{2} \\ \frac{I^{fc}}{\Delta t_n} - \frac{A_n^{fc}}{2} & \frac{I^{cc}}{\Delta t_n} - \frac{A_n^{cc}}{2} \end{pmatrix} \begin{pmatrix} w_{n+1}^f \\ p_{n+1} \cdot w_{m\cdot 2^l}^c + (1-p_{n+1}) \cdot w_{(m+1)\cdot 2^l}^c \end{pmatrix} \end{aligned} \tag{2.28}$$

or

$$\begin{aligned}(Bw)_n = &\\ &\begin{pmatrix} p_n\left(-\frac{I^{cf}}{\Delta t_n} - \frac{A_n^{cf}}{2}\right) & -\frac{I^{ff}}{\Delta t_n} - \frac{A_n^{ff}}{2} & (1-p_n)\left(-\frac{I^{cf}}{\Delta t_n} - \frac{A_n^{cf}}{2}\right) \\ p_n\left(-\frac{I^{cc}}{\Delta t_n} - \frac{A_n^{cc}}{2}\right) & -\frac{I^{fc}}{\Delta t_n} - \frac{A_n^{fc}}{2} & (1-p_n)\left(-\frac{I^{cc}}{\Delta t_n} - \frac{A_n^{cc}}{2}\right) \end{pmatrix} \cdot \begin{pmatrix} w_{m\cdot 2^l}^c \\ w_n^f \\ w_{(m+1)\cdot 2^l}^c \end{pmatrix} \\ + &\begin{pmatrix} p_{n+1}\left(\frac{I^{cf}}{\Delta t_{n+1}} - \frac{A_{n+1}^{cf}}{2}\right) & \frac{I^{ff}}{\Delta t_{n+1}} - \frac{A_{n+1}^{ff}}{2} & (1-p_{n+1})\left(\frac{I^{cf}}{\Delta t_{n+1}} - \frac{A_{n+1}^{cf}}{2}\right) \\ p_{n+1}\left(\frac{I^{cc}}{\Delta t_{n+1}} - \frac{A_{n+1}^{cc}}{2}\right) & \frac{I^{fc}}{\Delta t_{n+1}} - \frac{A_{n+1}^{fc}}{2} & (1-p_{n+1})\left(\frac{I^{cc}}{\Delta t_{n+1}} - \frac{A_{n+1}^{cc}}{2}\right) \end{pmatrix} \cdot \\ &\qquad \begin{pmatrix} w_{m\cdot 2^l}^c \\ w_{n+1}^f \\ w_{(m+1)\cdot 2^l}^c \end{pmatrix} \end{aligned} \tag{2.29}$$

2.2. *Reduction in computational complexity*

For the multirate time-stepping method, the system is described by a $2^{-l}N \times 2^{-l}N$ symmetric tridiagonal matrix of block size $(2^l|\Omega_1| + |\Omega_2|) \times (2^l|\Omega_1| + |\Omega_2|)$, making the computational cost of solving (2.25) using multigrid method equal to

$$6C_{mv}(2^l|\Omega_f| + |\Omega_c|)n_{nz}(l)n_j\frac{2^{-l}N}{P(l)} + 2\beta n_j \log((2^l|\Omega_f| + |\Omega_c|)2^{-l}NP(l)),$$

where C_{mv} is the amortized time per floating point operation for sparse matrix-vector multiplication, β is the average time to transmit one floating point number between any two processing elements across the network, n_{nz} is the average number of nonzero entries per row of A_i, and n_j is the number of pre- and post-smoothing Jacobi-like iterations. P is the number of processors.

Note that n_{nz} and P are the values that depend on l. The dependence of n_{nz} to l depends on the stencil size and the interface size of the two domains Ω_f and Ω_c; however, compared to the entire domain, we are assuming these two values are small, so we can assume n_{nz} to be constant. More detailed analysis of n_{nz} will follow in the next section. We can optimize the value of P such that the computational cost is minimized. Such value of P is

$$P(l) = 3\frac{C_{mv}n_{nz}(2^l|\Omega_f| + |\Omega_c|) \cdot 2^{-l}N}{\beta}.$$

The minimum computational cost is therefore

$$2\beta n_j \left[1 + \log\left(3\frac{C_{mv}n_{nz}}{\beta}\right) + 2\log\left(2^{-l}N(2^l|\Omega_f| + |\Omega_c|)\right)\right].$$

In the case $l = 0$, thus the traditional LSS method, this computational cost is

$$2\beta n_j \left[1 + \log\left(3\frac{C_{mv}n_{nz}}{\beta}\right) + 2\log\left(N(|\Omega_f| + |\Omega_c|)\right)\right].$$

Then, the speed up can be written as

$$1 \Bigg/ \left\{1 - \frac{2\log\left[(|\Omega_f| + |\Omega_c|) \,/\, \left(|\Omega_f| + 2^{-l}|\Omega_c|\right)\right]}{1 + \log\left(3C_{mv}n_{nz}/\beta\right) + 2\log\left[N(|\Omega_f| + |\Omega_c|)\right]}\right\}.$$

This is not a positive speed up in all cases, but for cases with $|\Omega_f| \ll |\Omega_c|$ and $l \gg 1$, which are the cases of interest, the speed up can be significant as seen in the next section.

2.3. *Constant memory requirement*

MTLSS has the same memory requirement per core compared to the single rate LSS. We see from the above setup that the most storage demand comes from forming the block diagonal matrix B which is a $2^{-l}N \times 2^{-l}N$ matrix with block size $(2^l|\Omega_1| + |\Omega_2|) \times (2^l|\Omega_1| + |\Omega_2|)$.

From the definition of B, we see that the nonzeros in row i of a block in matrix B correspond to the points in the stencil of x_i. (This holds for A_i as well.) Since B is block bi-diagonal, B only has $2n_s$ nonzeros in each row, where n_s is the number of points in the stencil. E has only one nonzero element per row. We also store u, w and b.

If we use the number of processors that minimize computational cost according to the previous section, each processor houses

$$\frac{(2n_s + 4)\beta}{3C_{mv}n_{nz}}$$

floating points. Since $n_{nz} = n_s$, we can rewrite this as

$$\frac{(2n_s)\beta}{3C_{mv}n_s} = \frac{2\beta}{3C_{mv}} + \frac{4\beta}{2C_{mv}n_s}$$

doubles.

As we go from a single-rate LSS to MTLSS, n_s increases for those points on the boundary of Ω_f and Ω_c. However, this increase is in a small subdomain of the entire spatial domain, and the increase in the number of elements in the stencil decreases the memory required.

3. Applications

3.1. *Stochastic Burgers' equation*

First, consider the Burgers' equation subject to random forcing with no slip boundary condition in nondimensional form

$$\begin{cases} u_t = -uu_x + \frac{1}{\mathrm{Re}}u_{xx} + \chi(x,t),\ 0 < x < 1, \\ u(x=0) = u(x=1) = 0, \end{cases} \tag{3.1}$$

where u is the velocity, $\mathrm{Re} = UL/\nu$ is the Reynolds number, χ is the random forcing and

$$< \chi >_x = 0, \quad < \chi^2 >_x = 1. \tag{3.2}$$

Here $< \cdot >_x$ denotes the average value over space. Define $\chi_i^n = \chi(x_i, t_n)$. At each instant of time t_n, the χ_i^n are uncorrelated random variables in space; for each fixed i, χ_i^n is constant on a time interval $(t_{pr}, t_{(p+1)r})$, where r and p are integers and r denotes the interval that the χ_i^n's are constant. The initial condition is $u(x,0) = 0$. The spatial grid consisted of 555 points with spacing of $\frac{1}{512}$ in the interior and $\frac{1}{2048}$ near the boundaries following the set up of Chambers *et al.* (1988). We define the domain with spacing $\frac{1}{512}$ as Ω_c and the domain with spacing $\frac{1}{2048}$ as Ω_f.

As our reference solution, u_{ref}, we use the results from a fourth-order Runge Kutta (RK-4) method in time and second-order in space solution with parameters $\mathrm{Re} = 500$, $\Delta t = 0.001$ and $r = 100$. We then apply MTLSS to find the solution for $\mathrm{Re} = 1500$ for the same time parameters. We plot in Figure 1 the solution obtained from MTLSS and the solution from a RK-4 method in time and second order in space of the $\mathrm{Re} = 500$ (u_{ref}) and $\mathrm{Re} = 1500$ case. All three solutions are averaged over 25000 time-steps.

In Chambers *et al.* (1988) and Choi *et al.* (1993), it is observed that the wall-layer thickness gets thinner as the Reynolds number increases due to the convective nature of the solution of the Burgers' equation. In the solution obtained by MTLSS, we observe that the wall-layer thickness δ is the same as the wall-layer thickness of the RK-4 solution at $\mathrm{Re} = 1500$ with $\delta = 0.9570$, compared to the wall-layer thickness of the RK-4 solution at $\mathrm{Re} = 500$ of $\delta = 0.9697$.

3.2. *Two-Dimensional Burgers' equation*

Consider the two-dimensional Burgers' equation.

$$u_t = -uu_x + \nu(u_{xx} + u_{yy}) \tag{3.3}$$

subject to the initial condition

$$u(x,y,0) = \exp[-(x^2 + y^2)], (x,y) \in \Omega$$

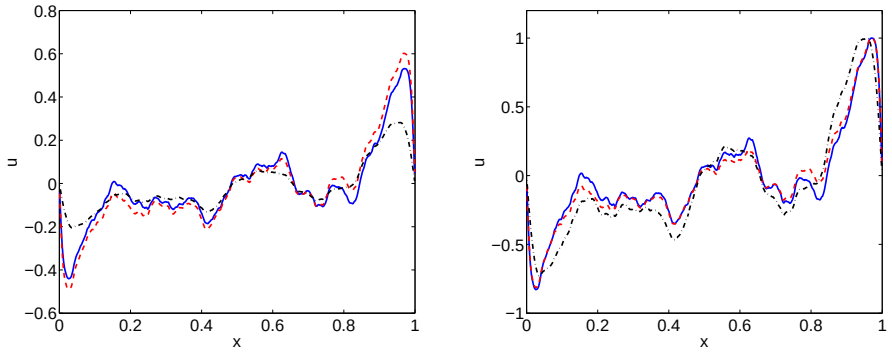

FIGURE 1. Time averaged solution and normalized by largest velocity solution for equation (3.1). MTLSS, Re = 1500 ——; RK-4, Re = 1500 - - -; RK-4, Re = 500 - · -.

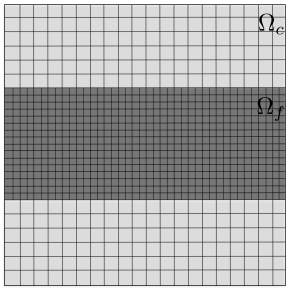

FIGURE 2. Grid used in the 2D Burgers' equation. Each cell represents 4 cells in the actual grid. The shock profile moves left to right inside Ω_f.

and periodic boundary conditions, where $\Omega = \{(x, y)| - 4 \leq x, y \leq 4\}$.

We solve this problem numerically using uniform cartisian grid. We first generate a reference solution u_{ref} using a small Reynolds number, $\nu = 0.25$. We then refine the uniform grid on a subdomain by interpolating the reference solution and use MTLSS in order to generate a solution for a high Reynolds number, $\nu = 0.025$.

The solution for the smaller Re has subdomains where the shock forms that require finer meshing than other parts of the domain. We can divide the domain so that the parts affected by the shock have smaller grid size, as shown in Figure 2. Due to the difference in grid size along with the difference in velocity, in order to insure stability, it is necessary to take smaller time-steps in this region. We then apply MTLSS to this problem. The results are given in Figure 3.

4. Summary and further plans

We have shown that the multirate time-stepping least squares shadowing method has computational advantage over the traditional least squares shadowing method with the same memory requirements. In cases where the global time-step is restricted by a small domain in space, this improvement is significant. We have shown through Burgers' equation in 1D and 2D, that the MTLSS is effective in finding statistics of an ergodic system, and that it is as effective as the LSS in obtaining the solution. Our future plans for

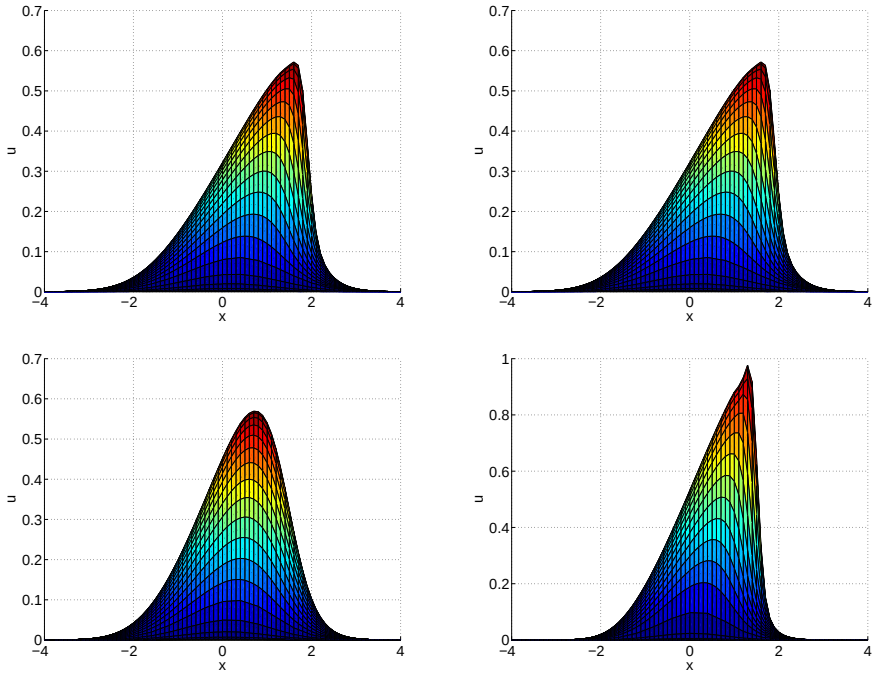

FIGURE 3. Solution for Eq. (3.3). MTLSS, $\nu = 0.025$ (top, left); LSS, $\nu = 0.025$ (top, right); RK-4, $\nu = 0.25$ (u_{ref})(bottom, left); RK-4, $\nu = 0.025$ (bottom, right).

MTLSS is to apply this method to a fully turbulent 3D flow. We are also looking at ways to improve temporal accuracy of this method.

Acknowledgments

The authors are grateful for the very valuable discussions on the subject with Q. Wang and M. Minion. Support from Stanford Graduate Fellowship is gratefully acknowledged.

REFERENCES

CHAMBERS, D. H., ADRIAN, R. J., MOIN, P., STEWART, D. S. & SUNG, H. J. 1988 Karhunen-Loéve expansion of Burgers' model of turbulence. *Phys. Fluids* **31**, 2573–2582.

CHOI, H., TEMAM, R., MOIN, P. & KIM, J. 1993 Feedback control for unsteady flow and its application to the stochastic Burgers' equation. *J. Fluid Mech.* **253**, 509–543.

CONSTANTINESCU, E. & SANDU, A. 2007 Multirate timestepping methods for hyperbolic conservation laws. *J. Sci. Comp.* **33**, 239–278.

PILYUGIN, S. 1999 *Shadowing in Dynamical Systems*. Springer.

SENY, B., LAMBRECHTS, J., TOULORGE, T., LEGAT, V. & REMACLE, J.-F. 2014 An efficient parallel implementation of explicit multirate RungeKutta schemes for discontinuous Galerkin computations. *J. of Comp. Phys.* **256**, 135–160.

WANG, Q., GOMEZ, S. A., BLONIGAN, P. J., GREGORY, A. L. & QIAN, E. Y. 2013 Towards scalable parallel-in-time turbulent flow simulations. *Phys. Fluids* **25**, 110818.